MitM Attack Detection in BLE Networks using Reconstruction and Classification Machine Learning Techniques

Abdelkader Lahmadi, Alexis Duque, Nathan Heraief, Julien Francq

▶ **To cite this version:**

Abdelkader Lahmadi, Alexis Duque, Nathan Heraief, Julien Francq. MitM Attack Detection in BLE Networks using Reconstruction and Classification Machine Learning Techniques. MLCS 2020 - 2nd Workshop on Machine Learning for Cybersecurity, Sep 2020, Ghent, Belgium. pp.1-16. hal-02948407

HAL Id: hal-02948407
https://inria.hal.science/hal-02948407

Submitted on 24 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MitM Attack Detection in BLE Networks using Reconstruction and Classification Machine Learning Techniques *

Abdelkader Lahmadi[1], Alexis Duque[2], Nathan Heraief[3], and Julien Francq[3]

[1] Université de Lorraine, CNRS, Inria, Loria, F-54000 Nancy, France
lahmadi@loria.fr
[2] Rtone - 120 rue de Saint-Cyr - 69009 Lyon - France
alexis.duque@rtone.fr
[3] Airbus CyberSecurity SAS, 78996 Elancourt Cedex, France

Abstract. Internet of Things (IoT) devices, including smartphones and tablets, are widely deployed in various application domains ranging from smart homes to industrial environments. Many of these devices rely on Bluetooth Low Energy (BLE) as a communication protocol for their control or the transfer of data. Trivial attacks can easily target these devices to compromise them due to their low security features and inherent vulnerabilities in their software and communication components. In this paper, we firstly demonstrate a Man-in-the-Middle (MitM) attack against BLE devices while collecting datasets of network traffic data exchange with and without the attack. Secondly, we study the use of machine learning to detect this attack by combining unsupervised and supervised techniques. We applied and compared two unsupervised techniques to reconstruct the model of BLE communications and detect suspicious data batches. We then applied a classification method based on Text-CNN technique to classify packets as normal or attack inside each suspicious batch. Our model reconstruction results show that we are able to discriminate normal and attack models with high precision and our classification method achieves high accuracy (≈ 0.99) and low false positive rate (≈ 0.03).

Keywords: IoT security · Bluetooth Low Energy · Neural networks · Machine learning · Attack detection

1 Introduction

BLE (Bluetooth Low Energy) is a widely used radio technology by connected devices including smartwatches, smartphones, smart plugs and smart locks, that are referred as Internet-of-Things (IoT). The number of these devices and their deployment in particular for smart homes and industrial environment is growing, and they are also becoming a subject of potential security issues. They

* This work is funded by the French Government under grant FUI 23 PACLIDO (Protocoles et Algorithmes Cryptographiques Légers pour l'Internet Des Objets).

are vulnerable even to trivial attacks and can be easily compromised due their limited security features and lacking of secure development practices. Multiple existing research works have shown real world discovered vulnerabilities [11] in BLE devices and demonstrated attacks against them [20]. An easy to deploy and perform attack on BLE devices is the Man-in-the-Middle (MitM) attack by using several available tools (BTLeJuice, GATTack, Mirage) and with low cost hardware (few BLE adapters) [7]. This attack could be performed even if the device is not too close to the attacker by abusing BLE-enabled smartphones or by remotely controlling a mobile malware. In addition, with special radio adapters and amplifiers, an attacker can intercept BLE signals up to 1,000 meters while initially the BLE radio range is up to 100 meters [20].

Plenty of work has been done to improve the security of IoT devices by providing detection and protection methods [18]. In particular, several work rely on machine learning techniques to identify anomalies in network traffic through offline or online analysis [8]. Nguyen *et al.* [13] proposed a machine learning based system for detecting compromised IoT devices. Their system uses devices specific communication patterns to detect anomalous behaviours deviation caused by attacks. They applied a federated learning approach using Deep Neural Networks (DNN) to train models locally and then update a centralised model. However, most of existing work focused on volumetric attacks, such as Mirai [2] and few and rare work interested in attacks with sporadic network activity such as MitM, in particular for BLE based systems [1]. Oliff *et al.* [14] proposed a detection method based on machine learning for spoofing attacks in BLE enabled occupancy system. Their method uses three classifiers with location labeled BLE advertising packets and under identity spoofing attacks. The proposed method is able to detect attack with an accuracy ranging from 80% to 91%. Zuo *et al.* [20] have shown that a large number of deployed BLE devices and their companion mobile app are easy to fingerprint and rely on "Just Works" pairing mode which allows attackers to hijack their connections using MitM attack. Yaseen *et al.* [19] addressed the issue of detecting MitM in BLE based eHealth care systems by using anomaly detection metrics.

In this paper, we propose a machine learning based method for detecting MitM attacks by using datasets of a concrete attack scenario on BLE devices. Our method relies on reconstruction and classification techniques to detect suspicious network data batches that have large deviations from benign patterns of behaviour and then detect inside each of them attack packets. For the reconstruction technique, we compared the performance of Long Short-Term Memory (LSTM) and Temporal Convolutional Network (TCN) based auto-encoders to learn normal models of BLE packets. Our results show that a TCN approach is more accurate and provides higher temporal memory effect since our datasets are of small size. The classification technique combines payload bytes embedding and statistical features to learn, by using a Convolutional Neural Network (CNN) architecture, latent features of packets and in a second stage we use a Random Forest algorithm to classify packets. By combining the two techniques,

we were able to detect and classify the BLE packets with high accuracy (≈ 0.99) and low false positive rate (≈ 0.03).

The rest of the paper is organised as follows. In section 2, we provide an overview of the BLE protocol and its main features, packets and operations. In section 3, we detail our experimental set-up of the MitM attack and the process of collecting the datasets. In section 4, we describe our ML-based detection method by jointly applying reconstruction and classification techniques. Section 5 concludes the paper and provides perspectives of this work.

2 BLE overview

Bluetooth Low Energy (BLE) was introduced by the Bluetooth Special Interest Group (SIG) in [15] as a variant targeted towards battery-powered Internet of Things (IoT) applications such as fitness trackers, headphones and smartwatches. BLE is becoming one of the most common wireless standards used today in IoT devices. According to the Bluetooth SIG, more than two billion devices supporting BLE have been shipped in 2018 [16]. Likewise, it is also becoming more commonly used in applications where sensitive information is being transferred.

2.1 BLE advertising and connection

BLE uses the same 2.4 GHz ISM band as Bluetooth Classic and Wi-Fi. The BLE specification divides the band into 40 channels of 1 MHz spaced 2 MHz apart. Three of these channels are called advertising and are used by devices exclusively to send beaconing packets called advertising packets.

BLE specification defines two roles: Peripheral and Central. Central devices are the ones that initiate connection, while Peripherals accept. In this way a Central devices acts as a master, on which many Peripheral slaves can be attached. Figure 1 provides an overview of BLE workflow between a peripheral and a central (smartphone). The Central device will listen for advertisements from Peripheral devices but once the advertisement from the desired Peripheral device is received, the Central may connect by entering the initiating state. For the Peripheral device, the advertising state is also the initial state before the connection state. The connection state is the final state in which the Peripheral and Central devices can exchange data.

The BLE Link Layer offers two mechanisms for exchanging data in BLE: advertising and connections. Advertising allows sending unidirectional but broadcast data. The Peripheral device sends data using Advertising and Scan Response packets. Because it is broadcast in nature, multiple devices can listen to the advertising data. Each advertising packet is configurable by the product developer and can contain a wealth of information. It is not necessary to connect to a device to get these packets, but the Central cannot send any data back. Connections allow the Central and Peripheral to exchange data bidirectionally, controlling the device and sending it information, as opposed to the unidirectional nature of advertising. So advertising packets serve dual roles: they enable Central devices to find devices and connect, and also able to convey information.

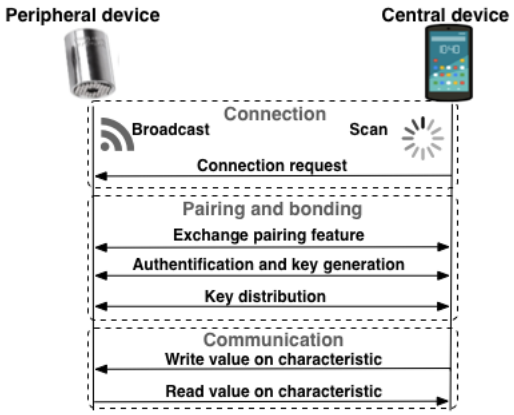


Fig. 1: BLE workflow between peripheral and central devices.

2.2 Data exchange

Data is transmitted on 37 data channels which are not used for advertising. When devices are in a connection, they periodically exchange packets during connection events. The rate of these events is defined by parameters such as *Connection Interval*. The BLE specification allows the peripheral device to skip connection events if there are no data to exchange.

The logical link control and adaptation protocol (L2CAP) within the BLE stack fragments and re-assembles packets from other layers. It takes packets received by the Link Layer and forwards them the Generic Attribute (GATT) protocol for accessing data or the Security Manager. Data exposed by a Peripheral are presented in a GATT profile which is a hierarchical structure of attributes allowing the transfer of information between a Central device and a Peripheral device. Within a GATT profile, attributes can be either services or characteristics and are identified by a universally unique identifier (UUID). In addition to their UUID, characteristics are made up of an attribute handle, a set of properties and a value. The handle specifies the position of the characteristic in the profile and the value holds the actual data of the characteristic. Properties specify which operations (read, write, etc.) can be executed on each particular attribute and with which specific security requirements (encryption, authentication).

2.3 BLE security

A BLE connection is said to operate at a specific security mode for which there are several security levels. The required security mode and level of a connection may change from time to time, leading to procedures to increase that level. When two devices which initially do not have security, wish to do something which requires security, the devices must pair first. This process is triggered (for

example) by a Central device (e.g. a smartphone) that is attempting to access a data value of a characteristic on a Peripheral device that requires authenticated access. Pairing involves authenticating the identity of two devices, encrypting the link using a Short-Term Key (STK), and then distributing Long-Term Keys (LTK) used for encryption. The LTK is saved for faster reconnection in the future, that is termed Bonding.

The security level of the connection is based on the method of pairing performed and this is selected based on the I/O capabilities of each device. The security level of any subsequent reconnection is based on the level achieved during the initial pairing. When pairing, the method chosen determines if the pairing performs a strong authentication or not. Unauthenticated pairing occurs in situations where the device could not authenticate itself, for example if it has no Input/Output (I/O) capabilities. Pairing involves authenticating the identity of the two devices to be paired, usually through a secret-sharing process. Once authenticated, the link is encrypted and keys distributed to allow security to be restarted on a reconnection much more quickly. If these keys are saved for a future time, the devices are said to be *Bonded*. A pairing procedure involves an exchange of Security Manager protocol packets to generate a temporary encryption key called the Short Term Key (STK). During the packet exchange, the two peers negotiate one of the following STK generation methods: 'Just Works' where the STK is generated on both sides, based on the packets exchanged in plain-text, 'Passkey Display' where one of the peers displays a randomly generated 6-digit passkey and the other side is asked to enter it, "Out of Band (OOB)' where additional data is transferred by means other than the BLE radio, such as another wireless technology like Near Field Communication (NFC), 'Numeric Comparison' (Low Energy Secure Connections Pairing) which is only available with BLE 4.2 and it uses an algorithm called Elliptic Curve Diffie-Hellman (ECDH) for key generation, and a new pairing procedure for the key exchange. However, many BLE devices rely on the Just Works pairing method which is insecure and the devices become vulnerable to MitM attacks.

3 Experimental set-up

In this work, we consider the scenario of a BLE-enabled torque wrench device controlled remotely by a user through an App running on a smartphone to adjust and calibrate with high precision the torque settings (angle and force) as depicted in Figure 1 of Section 2. In this case study, the attacker could be located nearby the device or it could act remotely by compromising the smartphone with a malware. In the second situation, attacks made by a malware could be broad, and they require the exploitation of specific vulnerabilities to the smartphone, its OS or the running App that controls the torque wrench. In this work, we focus more on nearby attackers that perform Man-in-the-Middle (MitM) attacks to connect, pair, read, and even write to the device. This attack does not require specific vulnerabilities. The only constraint is that the attacker has to be within

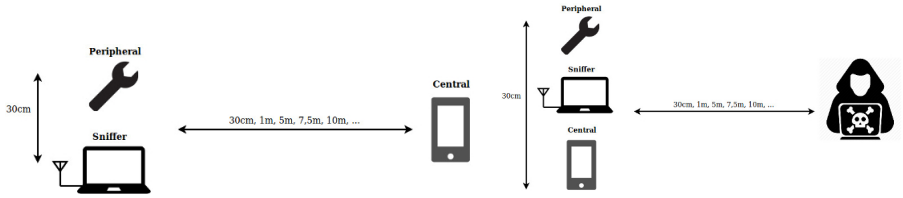
the communication range of BLE which is at most 100 meters which could be extended to 1,000 meters by using long range BLE sniffers [20].

We performed the MitM attack with a cloned BLE device identical to the torque wrench. The clone is realised by using 2 USB dongles Bluetooth 4.0 Cambridge Silicon Radio (CSR) and the Mirage tool [5]. The attacker uses this clone to read, modify and write the settings of the torque wrench which may alter its accuracy and the quality of operations. In particular, when the operator is adjusting the settings of the torque wrench with the desired values, the attacker will modify them and the applied torque will be different from the expected. In our experimental environment, we used the following devices and tools:

- Two devKits nRF52840. One is used by the torque wrench device, and the second is used as a sniffing interface for BLE packets.
- A smartphone with the nRFConnect installed and running to control remotely the torque wrench device.
- Two USB dongles to perform the MitM attack.
- Two hosts: one is used to perform the MitM attack by using the Mirage tool, and the second is running Wireshark tool for packets capture using the BLE sniffer.

3.1 Experimental Methodology

In our case study, we define two main scenarios. The first is a nominal scenario without any attack, and in a second scenario we introduced the MitM attack. For each scenario, using the experimental environment described above, we collect the BLE packets exchanged between the device and the smartphone while varying the distance between them. The distance as explained in Section 4 will be used as a feature for detecting the attack and allows us to measure the detection accuracy according to the closeness of the attacker to the device.



(a) Normal scenario (b) MitM scenario

Fig. 2: Experimental Set-up for normal and MitM scenarios.

Normal scenario. In this scenario, as depicted in Figure 2a, we collect the BLE packets exchanged between the BLE device and the App while performing these operations over time: from 0 to 1 minute the App reads 4 values, from 1

to 5 minutes the App activates and receives notifications, at 5 minutes the App deactivates the notifications, from 5 to 6 minutes the App writes 4 values, from 6 to 10 minutes the App activates and receives the notifications, at 10 minutes the App deactivates the notifications and reads the Device Name characteristic. These operations allows us to simulate a behaviour of the App running on the smartphone and generate different BLE packets including reading, writing and notifications.

MitM scenario. In this attack scenario, as depicted in Figure 2b, the attacker will modify values written by the smartphone App on the BLE device. From 5 to 6 minutes, the values written by the App are inverted by the attacker before being sent to the BLE device. From 6 to 10 minutes, the notifications sent by the BLE device to smartphone are modified. The timing of the modifications and the different operations are used for labelling the captured packets with "normal" and "attack" labels.

3.2 Datasets building

Using the two experiments described above, we collect different datasets of BLE packets exchanged between the device and the smartphone. We build multiple datasets by varying the distance between the smartphone and the BLE device for the normal scenario, and between the attacker and the smartphone for the attack scenario. As depicted in Figure 3, at time $t = 0$, the first seen packets are advertising messages. Then a connection is established between the smartphone and the device at $t = 2.792896$ with packet number 200.

No.	Time	Source	PHY	Protoco	Length	Delta time (μ	SN	NESN	More Da	Event cc	Info
1	0.000000	d2:57:fb:23:43:43	LE 1M	LE LL	21	46886				0	ADV_IND
2	0.000539	d2:57:fb:23:43:43	LE 1M	LE LL	21	497				0	ADV_IND
				....							
200	2.792896	77:87:db:4e:87:75	LE 1M	LE LL	34	150				0	CONNECT_REQ
201	2.893677	Master_0xb5b3ad99	LE 1M	LE LL	9	28851	0	0	False	0	Control Opcode: LL_FEATURE_REQ
202	2.894161	Slave_0xb5b3ad99	LE 1M	LE LL	0	150	0	1	False	0	Empty PDU
203	2.894514	Master_0xb5b3ad99	LE 1M	LE LL	0	44618	1	1	False	1	Empty PDU
204	2.894904	Slave_0xb5b3ad99	LE 1M	LE LL	9	151	1	0	False	1	Control Opcode: LL_FEATURE_RSP
205	2.995608	Master_0xb5b3ad99	LE 1M	LE LL	9	44617	0	0	False	2	Control Opcode: LL_LENGTH_REQ

Fig. 3: BLE advertising and connection packets exchanged between the device and the smartphone in a normal scenario with a distance of 1m between them.

We vary the distance between the devices in the set of values {30cm, 1m, 5m, 7.5m, 10m}. In total we obtain 10 datasets that we merge in a single dataset with the distance value as a feature and each sample is labeled as attack or normal. The obtained dataset has a size of 19MB and 77680 samples. We use 80% of this dataset for the training phase of ML algorithms and 20% for testing.

4 Detection Approach

Our detection approach of the attack described in section 3 relies on two machine learning techniques: reconstruction and classification. This first technique

aims at building a baseline model of normal patterns by using a machine learning algorithm and then we measure deviations and errors from that model [6]. Reconstruction is applied on batches of data and when one of them has a significative reconstruction error, it is considered as abnormal. The second technique applies a classification method to classify packets marked with attack features. The two techniques are applied jointly to detect respectively suspicious data and identify attacks in these suspicious packets. Figure 4 depicts the processing steps of our detection approach where reconstruction and classification techniques are applied jointly.

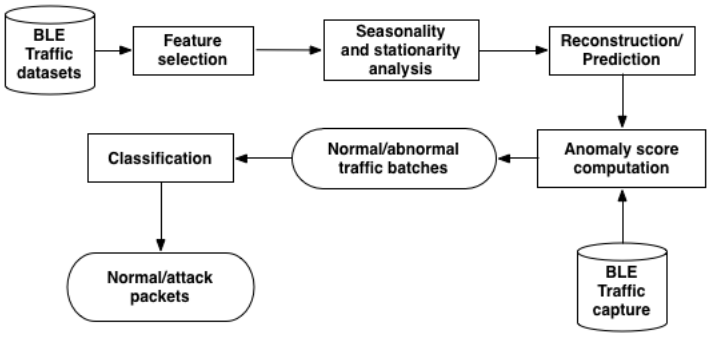


Fig. 4: Our MitM attack detection approach by using jointly reconstruction and classification techniques.

4.1 Features extraction and analysis

In this section, we describe our analysis of features extracted from BLE packets to select among them the most important for the reconstruction and classification techniques. We used the following 4 feature selection methods to identify an optimal set of features:

- Variance: this method applies a variance threshold to remove all low-variance features.
- Chi2: this method selects the features with the highest values of the chi-squared test.
- Recursive Feature Elimination (RFE): this method selects the features by recursively considering smaller and smaller sets of features while computing their importance.
- Extra trees: this method applies ensemble learning technique using decision tree provided with a random sample of k features and then select from them the most important features by using the Gini index.

Let $\mathbb{F} = (f_1, f_2, ..., f_n)$ the set of features extracted from a BLE packet, with $n = 250$. Each method i provides a subset of features $F_{k,i}$ composed of k features.

We apply then the intersection operator on the subsets F_k to obtain $\mathbb{F}_{final}$ with $\mathbb{F}_{final} = \bigcap_{i=1}^{4} F_{k,i}$. We applied the 4 methods while computing the performance of the machine learning algorithms until we obtain the optimal set of features. Our analysis shows that the following 4 features in a BLE packets dataset are the most important:

- Channel numbers: the channels used during the exchange of the BLE packets.
- Delta_time: the difference of time between two successive packets.
- Received Signal Strength Indication (RSSI): the signal-to-noise ratio value available in BLE packets.
- Distance: it denotes the distance between the mobile and the BLE device.

After selecting this set of optimal features, we analysed the stationarity and the seasonality of datasets. The stationarity denotes that the statistical properties of a feature are all constant over time. The seasonality denotes periodic patterns in the datasets that should be eliminated prior to building reconstruction models. To guarantee that our selected features represented as time series are stationary, we applied the following tests: Augmented Dickey-Fuller test, and Kwiatkowsk-Phillips-Schimdt-Shin test [4]. Our tests show that the 4 selected features are stationary and we eliminated from them the seasonality patterns.

4.2 LSTM based model reconstruction

The reconstruction method consists in learning the normal behaviour of the BLE packets exchange. In the training phase we are looking to minimise the error between the learned data and the original dataset. In the testing phase, if the data contains an abnormal behaviour, the reconstruction will decrease which allows us to detect such behaviour in the observed packets. A technique to realise a model construction is to rely on a neural network of type Long Short-Term Memory (LSTM) [9]. In this way, we are able to approximate the BLE applications behaviour while considering their temporal patterns. By using this technique, we applied the following steps:

- Train the neural network on the dataset X_{train};
- Evaluate the obtained model on the $X_{validation}$ part while computing the reconstruction error;
- Set a detection threshold to determine the presence of anomalies. We can set, for instance, this threshold to 3 standard deviations of the mean value of the error, which is an empirical choice and a widely used threshold value for anomaly and outliers detection.

We realised this technique by using an LSTM auto-encoder which is a neural network with an Encode-Decoder LSTM architecture [17]. The hyperparameters of the used LSTM neural network are presented in Table 1. To set the detection threshold, we compute the residual defined as $R(X, \widehat{X}) = |X - \widehat{X}|$ with $\widehat{X} = f(X)$ and f represents the transformation of our auto-encoder. At the end of the

Table 1: The hyperparameters of the LSTM neural network.

Hyperparameter	Value
Optimizer	Adam
Learning rate	0.001
Batch size	40
Epoch number	150
Loss function	MSE
Validation metric	Accuracy
Validation split	0.2
DL framework	Tensorflow 1.13.1, Keras 2.2.4, Keras-tcn 2.6.7

training phase, we compute the mean and the standard-deviation of the residual $R(X_{train}, \widehat{X_{train}})$.

In the testing phase, we evaluate the residual $R(X_{test}, \widehat{X_{test}})$ to determine for each data batch its anomaly score α defined as following:

$$\alpha = \begin{cases} 0 \text{ if } |R(X_{test}, \widehat{X_{test}}) - \mu(R(X_{train}, \widehat{X_{train}}))| \leq 3 * \sigma(R(X_{train}, \widehat{X_{train}})) \\ 1 \text{ otherwise.} \end{cases}$$

We used the score α to detect the presence of an anomaly in a data batch if the residual is greater than 3 standard-deviations of the average. We obtain thus $X_{train} \in \mathcal{M}_{T,F}(\mathbb{R})$ which is the dataset of the nominal communication patterns of BLE with $T = 28918$ the number of samples, and $F = 4$ the number of features. We obtain in total: $T * F = 115672$ values. Then, as shown in Figure 5, we convert our training dataset to a tensor $T^{s,t,F}$ with $s = 4819$ the number of samples, $t = 6$ the time-step (*empirical choice*) and $F = 4$ the number of features. Indeed, we obtain in total $s * t * F = T * F$.

In a first step, during the training phase we build a normal model of BLE communications from a subset of the training dataset. If the model is close to the normal behaviour of these communications, the reconstruction error should be low. Figure 6a shows the details of this phase. We observe that the real values fit well the predicted values with a very low reconstruction error (close to 0).

In a second step, we tested the reconstruction model obtained from X_{train} on X_{test} which contains the MitM attack packets. Figure 6b shows the details of the reconstruction of the attack model. We clearly observe large reconstruction errors and the testing data does not fit with the training model which indicates the presence of anomalies.

To measure the reconstruction error between the normal and attack models, we compared several error metrics which are BIAS (Bias of an estimator) , MSE (Mean Squared Error), RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) [3]. The construction errors using these metrics are shown in Figure 7. We observe that the metric MSE measures with high precision the reconstruction error of the MitM attack traffic from the normal traffic. We observe

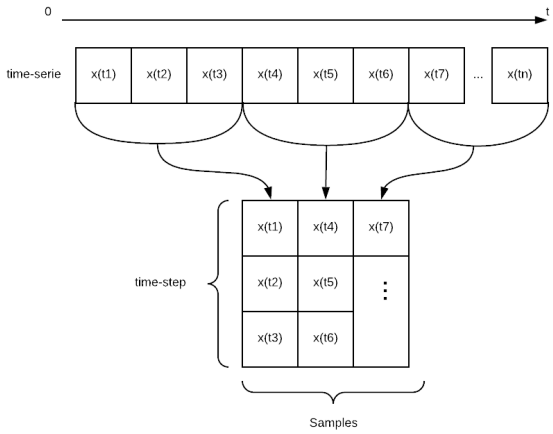
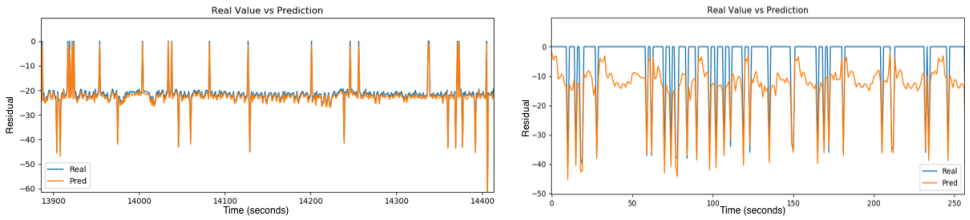


Fig. 5: Transformation of time series into training samples with a time-step = 3.



(a) Training of the normal model (b) Testing of the MitM attack model

Fig. 6: Training and testing of models reconstruction using LSTM.

also that all the metrics provide a low reconstruction error when predicting the same normal traffic as input.

A major drawback when applying the LSTM based auto-encoder technique in our case is the low value of the time-step, which is equal to 6 used for building the input sequences. We have thus a low memory effect in the training neural network. We can hardly increase this value, since our dataset is small and the training phase will face the gradient vanishing or exploding problem where the gradient becomes vanishingly small which prevents the neural network weights from changing their values during the training phase with.

4.3 TCN based model reconstruction

Another technique for model reconstruction is using a Temporal Convolutional Network (TCN) [10] instead of a LSTM based auto-encoder. A TCN is a class of time-series model that employs a hierarchy of temporal convolutional with an encoder-decoder architecture. It has the advantage of obtaining better results

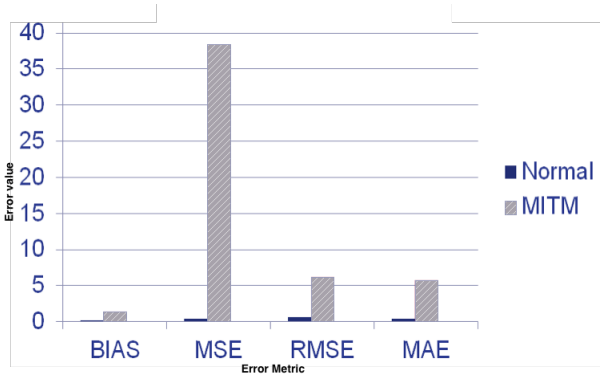
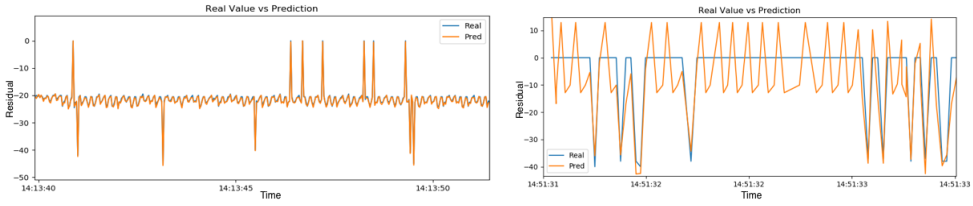


Fig. 7: Reconstruction error between normal and attack patterns using LSTM.

with less samples which allows us to increase the time-step when building the input sequences. The hyperparameter values of the TCN neural network are similar to those used for the LSTM neural network, as presented in Table 1.

As input to the TCN, we used the $X_{train} \in \mathcal{M}_{T,F}(\mathbb{R})$ dataset which contains normal communication patterns of BLE, with $T = 28918$ the number of temporal samples and $F = 4$ the number of features. We convert these time series to a tensor $T^{s,t,F}$ with $s = 963$ the number of samples, $t = 30$ the time-step and $F = 4$ the number of features. Using this model the time-step is 5 times higher than the LSTM based model. The prediction results of the normal behaviour of BLE communications using TCN are depicted in Figure 8a. We mainly observe that the predicted values fit well the real values and the reconstruction error is close to 0.



(a) Training of the normal model (b) Testing of the MitM attack model

Fig. 8: Training and testing of models reconstruction using TCN.

The results of the testing phase by comparing the real and predicted values are shown in Figure 8b. Similar to the LSTM based models, we observe that the predicted values do not fit the real values and we obtain large reconstruction

errors. The reconstruction errors using different metrics when comparing normal and attack enabled BLE traffic are depicted in Figure 9. The TCN model has more accurate and lower reconstruction error with high memory effect compared to LSTM architecture (Figure 7) when predicting attack traffic behaviour from normal traffic behaviour. Using the same anomaly score α, we are able to discriminate data batches containing suspicious packets. However, both LSTM and TCN models are only able to detect suspicious batches, without detecting packets involved in the attack.

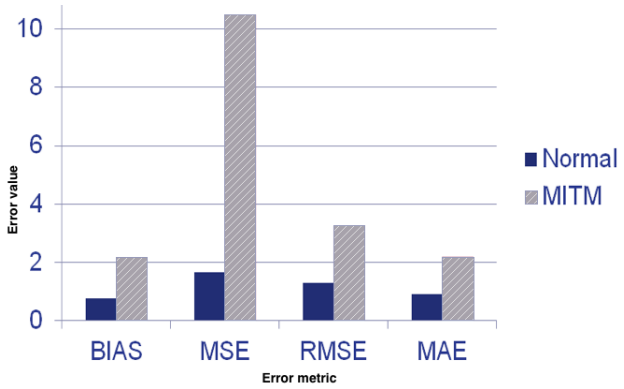


Fig. 9: Reconstruction error between normal and attack patterns using TCN.

4.4 Classification of BLE packets

After detecting suspicious batches of traffic with attack packets, the next step of our detection process is to classify these packets according to their class: "normal" or "attack". In our work, we applied the technique developed in [12] by jointly using Text-Convolutional Neural Network (Text-CNN) for feature extraction and a Random Forest algorithm for classification. In [12], the authors show that combining Text-CNN for payload feature extraction and a Random Forest algorithm for final packets classification outperforms a CNN model with a *softmax* classifier. For BLE packets available in the dataset, we extract from them their traffic statistics and we convert their payload into word embedding to extract salient features with Text-CNN. The hyperparameters of the Text-CNN neural network are presented in Table 2.

The statistical features that we extracted from the BLE traffic data are presented in Table 3.

The payload based features are extracted by converting packet payload bytes to low dimensional vectors using Word2Vec technique and then provide these vectors as an input to a Text-CNN neural network. The extracted features are

Table 2: The hyperparameters of the Text-CNN neural network.

Hyperparameter	Value
Optimizer	Adam
Learning rate	0.0001
Batch size	50
Epoch number	50
Loss function	binary cross-entropy
Validation metric	Accuracy
Validation split	0.2
DL framework	Tensorflow 1.13.1, Keras 2.2.4, Gensim (Word2Vec) 3.7.1

Table 3: Statistical features of BLE traffic data.

Features
Number of packets per second
Number of bytes per second
Max, min and average packets length
Max, min and average time interval between 2 packets
Number of packets for each BLE packets type (ADV, DATA, etc.)

then concatenated with the statistical features and provided as input to a Random Forest algorithm using a number of estimators equal to 200 to classify the packets. Our classification results are shown in the confusion matrix of Table 4.

Table 4: Confusion matrix of the classification of BLE packets.

		Predicted labels	
		Normal	Attack
Actual label	Normal	100% (9541/9543)	0% (2)
	Attack	0.3% (12)	99.7% (4207/4219)

We observe that mostly all the packets are classified correctly with only 2 normal packets misclassified as attack and 12 attack packets misclassified as normal.

The ROC curve of our classifier is depicted in Figure 10 with an Area Under the Curve (AUC) close to 1 which confirms that our classification model has a good measurement of separability between "normal" and "attack" packets. However, we have to note that our obtained results with high classification performance, are limited to the collected datasets within our experimental setup environment, and it is still difficult to generalise the learned models on new datasets with different settings.

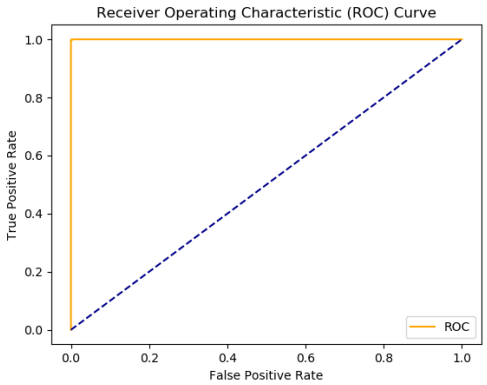


Fig. 10: ROC curve of the BLE packets classifier.

5 Conclusion and future work

In this paper, we presented a study on the use of machine learning techniques to detect MitM attack targeting BLE enabled IoT devices. This attack is trivial to deploy and may be used easily by attackers to stole users private information or alter control data exchanged between a companion mobile app and the device. We demonstrated the feasibility of the attack in a real-world deployment while varying the distance between the BLE mobile and devices and collecting datasets of exchanged BLE packets. We applied jointly reconstruction and classification models based on neural networks to detect suspicious network data traffic batches and then identify from them attack packets. Our evaluation results show high detection accuracy (≈ 0.99) and low false positive rate (≈ 0.03). In future work, we will extend the proposed method for detecting more classes of BLE attacks including DoS and connection hijacking within various BLE environments. We will also study efficient protection mechanisms for BLE networks.

References

1. Albahar, M., Haataja, K., Toivanen, P.: Bluetooth MITM Vulnerabilities: A Literature Review, Novel Attack Scenarios, Novel Countermeasures, and Lessons Learned. International Journal on Information Technologies & Security, **8** (December 2016)
2. Antonakakis, M., et al.: Understanding the Mirai Botnet. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1093–1110 (Aug 2017)
3. Botchkarev, A.: Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology (2018)
4. Box, G.E.P., Jenkins, G.M.: Time Series Analysis: Forecasting and Control. Prentice Hall PTR, USA, 3rd edn. (1994)

5. Cayre, R., Roux, J., Alata, E., Nicomette, V., Auriol, G.: Mirage : un framework offensif pour l'audit du Bluetooth Low Energy. In: Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC 2019)
6. Friedman, E., Dunning, T.: Practical Machine Learning: A New Look at Anomaly Detection (2014)
7. Goyal, R., Dragoni, N., Spognardi, A.: Mind the Tracker You Wear: A Security Analysis of Wearable Health Trackers. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 131–136. SAC '16 (2016)
8. Hafeez, I., Antikainen, M., Ding, A.Y., Tarkoma, S.: IoT-KEEPER: Detecting Malicious IoT Network Activity Using Online Traffic Analysis at the Edge. IEEE Transactions on Network and Service Management **17**(1), 45–59 (March 2020)
9. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997)
10. Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal Convolutional Networks for Action Segmentation and Detection **abs/1611.05267** (2016), http://arxiv.org/abs/1611.05267
11. Matheus E. Garbelini, Sudipta Chattopadhyay, C.W.: SweynTooth: Unleashing Mayhem over Bluetooth Low Energy. Tech. rep., Singapore University of Technology and Design (2020)
12. Min, E., Long, J., Liu, Q., Cui, J., Chen, W.: TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest. Security and Communication Networks **2018**, 1–9 (Jul 2018)
13. Nguyen, T., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., Sadeghi, A.: DIoT: A Federated Self-learning Anomaly Detection System for IoT. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)
14. Oliff, W., Filippoupolitis, A., Loukas, G.: Impact Evaluation and Detection of Malicious Spoofing Attacks on BLE Based Occupancy Detection Systems. In: Proceedings of the 1st International Conference on Internet of Things and Machine Learning. IML '17, ACM (2017)
15. SIG, B.: Bluetooth Core Specification 4.0 (Dec 2010), https://www.bluetooth.com/specifications/bluetooth-core-specification/
16. SIG, B.: Bluetooth Market Update 2019 (2019), https://www.bluetooth.com/wp-content/uploads/2018/04/2019-Bluetooth-Market-Update.pdf
17. Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised Learning of Video Representations Using LSTMs. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. pp. 843–852. ICML'15, JMLR.org (2015)
18. Vasilomanolakis, E., Daubert, J., Luthra, M., Gazis, V., Wiesmaier, A., Kikiras, P.: On the Security and Privacy of Internet of Things Architectures and Systems. In: 2015 International Workshop on Secure Internet of Things (SIoT)
19. Yaseen, M., Iqbal, W., Rashid, I., Abbas, H., Mohsin, M., Saleem, K., Bangash, Y.A.: MARC: A novel framework for detecting MITM attacks in ehealthcare BLE systems. J. Medical Systems **43**(11), 324:1–324:18 (2019)
20. Zuo, C., Wen, H., Lin, Z., Zhang, Y.: Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security